World Engineering & Applied Sciences Journal 8 (4): 153-160, 2017
ISSN 2079-2204
© IDOSI Publications, 2017
DOI: 10.5829/idosi.weasj.2017.153.160

Secure Authentication System in Cloud Encrypted Outsourced Data with Accurateness Enhancement

[1]P. Reshma and [2]N. Sudha

[1]PG Scholar, Computer Science and Engineering,
CMS College of Engineering and Technology, Coimbatore, Tamil Nadu, India
[2]Professor Computer Science and Engineering,
CMS College of Engineering and Technology, Coimbatore, Tamil Nadu, India

Abstract: In the current cloud computing scenario keyword-based search over encrypted outsourced data has become a significant method. The existent features aim on multi-keyword exact match or single keyword fuzzy search. However, being techniques have less possible significance in real-world applications compared with the multi-keyword fuzzy search technique over encrypted data. The first effort to build such a multi-keyword fuzzy search scheme was reported by Wang *et al*, used locality-sensitive hashing functions and Bloom filtering to meet the goal of multi-keyword fuzzy search. But, Wang's strategy was only useful for a one letter mistake in keyword but was not effective for other common spelling mistakes and Wang's scheme was vulnerable to server out-of-order problems during the ranking process and did not consider the keyword weight. Based on Wang *et al.*'s strategy, we propose an efficient multi-keyword fuzzy ranked search scheme based on Wang *et al.*'s concepts. Here, we are implementing a new way of keyword transformation based on the uni-gram, which will simultaneously focus on the accuracy and improves the ability to handle other spelling mistakes, along with keywords with the same root can be queried using the stemming algorithm. We consider the keyword weight when selecting an adequate matching file set.

Key words: Outsourcing security · Privacy-carry on · Approachable encryption · Multi-keyword search · Fuzzy search

INTRODUCTION

Besides, in order to raise practicability and preserve on the expense in the cloud paradigm, it is preferred to get the accessed result with the most relevant files that match users' interest instead of whole files, which values that the files should be ranked in the order of relevance by users' interest and only the files with the maximum relevance's are sent back to users.

A series of searchable symmetric encryption schemes are proposed to enable search on cipher text. Traditional SSE strategy points users to securely retrieve the cipher text. But these strategies fulfill only Boolean keyword search. Whether a keyword exists in a file or not without include the distinctness of relevance with the queried keyword of these files in the result. Saving the cloud from involving in ranking and entrusting all the work to the user is a natural way to eliminate information leakage. However, the limited computational power on the user side and the maximum computational overhead precludes information security.

Considering the flexibility and economic savings offered by the cloud server, the users are impelled to outsource the management of their data to the cloud. Because of the privacy matters, data owners encrypt private data prior to outsourcing, which makes data usage makes a challenging problem. Thus, development of an efficient privacy preserving search system over encrypted cloud data is of great challenge. The most common search method get files using keywords instead of accessing all the encrypted files back. To securely locating over encrypted data, the data owner commonly makes an encrypted index structure using the extracted keywords from the data files and a corresponding index-based

Corresponding Author: P. Reshma, PG Scholar, Computer Science and Engineering, CMS College of Engineering and Technology, Coimbatore, Tamil Nadu, India.

keyword [1] matching algorithm and subsequently outsources both the encrypted data and this constructed index structure to the cloud.

While searching the files, the cloud server combines the trapdoors of the keywords with the index information and finally returns the corresponding files to the data users. The data owner can share their data with many users which needs the cloud server to can meet many requests with effective data retrieval services. One of the optimal method for solving this problem is ranking [2] the results and sending back the top-*K* files to the data user instead of all the relevant files. This method can normally reduce the communication overhead and still meet user's demand. However, such a ranking operation should not leak any other information about the keywords.

In recent years, many attempts have been directed toward the design of efficient mechanisms for searching over encrypted data. Mostly these methods only make support to single keyword search and others simply offered conjunctive or disjunctive searches for multi-keyword queries. We observe that these techniques encourage only exact keyword matching. Thus, if keywords are incorrect or having spelling mistakes, results might be returned. Only a few skills have strike the fuzzy keyword search [3].

Wang *et al.*'s work was one of the first works to label the problem of multi-keyword fuzzy search [4] over encrypted data problem and did not specify a predefined fuzzy set (referred to as MFSE). In MFSE, a keyword was first make over into a bi-gram set and the Euclidean distance was used to capture keywords sameness. The MFSE subsequently used LSH functions from the same hash family to produce the Bloom filter based index and query. Due to the qualities of LSH, even if the keyword was incorrect, it could be hashed into the related bits in the query vectors with high possibility. Finally, this method used the inner product of the index vector and the query vector as the suitable score between queries and documents. This scheme resolve the problems of multi-keyword fuzzy search [4] with high efficiency and accuracy. The most important result was that their scheme does not require the predefined fuzzy set. However, some other issues become obvious in this scheme. First, transforming the keyword into a bi-gram set will increases the Euclidean distance. For example, for the spell out keyword "netward", the bi-gram set is *{ne, et, tw,wa, ar, rd}*. Two bigram sets are not similar compared with actual sets, which implies that the Euclidean distance between two vectors is 2 and the vectors are rarely be start into the same bit. Second, the scheme is not effective for other spelling mistakes. In fact, a keyword can be misspelled into many forms, not only one-letter errors. For example, the keyword "workout" can be spell out as "worktou" "wrokout" or "wokrout". All the above-specified spelling mistakes are common and should be considered by the search system. In addition, this scheme could not search the keywords with same root such as "make" and "making". Finally, MFSE did not find the relevance between the keywords and files. For the same keyword in different files, its keyword weight should be different and this difference should be contemplated during ranking. Thus, the files that are more applicable to the query keyword might not be inserted in the return results. To overcome the problems listed above, we develop a new multi-keyword fuzzy ranked search scheme based on MFSE. Our contributions can be summarized as follows:

- We develop a novel method of keyword transformation based on the uni-gram. For misspelling of one letter, this method slows up the Euclidean distance between the misspelled keyword and the correct keyword. Moreover, this method is also successful for other spelling mistakes. Additionally, we initiate the stemming algorithm to obtain the root of the word. Using this method, the keywords with the same root can also be enquired.
- We take the keyword weight into deliberation in constructing the ranked list of the results. The files that are more applicable to the keywords will have greater chances to notice first on the list.
- We implement and assess our proposed scheme using a real-world data set. The results reveal that our proposed scheme efficiently achieves high accuracy.

Problem Formulation: We work out the privacy problem of the multi-key word fuzzy ranked search over encrypted data in this section.

System Model: In this paper, we consider a cloud system made up of data owner, data user and cloud server, see Fig. 1. In our system model, data owner has a group of n data files F = (F1, F2, F3, • • •, Fn) and outsources them to the cloud server in the encrypted form C. To Formalize efficient search operation on these encrypted files, data owner will build a secure tractable index I on the keyword set W extracted from F. Both the index I and the encrypted data files C, are outsourced to the cloud server. To search the encrypted data files for given keywords, an authorized user integrates a corresponding trapdoor T and sends it to cloud server. Upon receiving the trapdoor, the cloud server is in charge to search the index I and return the

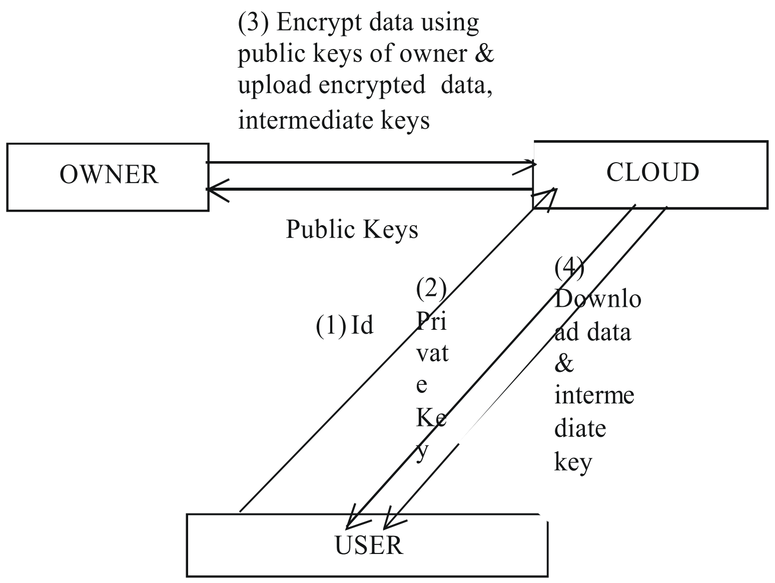

(3) Encrypt data using public keys of owner & upload encrypted data, intermediate keys
OWNER
CLOUD
Public Keys
(1) Id
(2) Private Key
(4) Download data & intermediate key
USER

Fig. 1: The system Architecture of our scheme.

correspondent set of the encrypted documents. To enhance the file saving quality and the communication cost, the search outcome should be ranked by the cloud server and get back the top-K relevant files to the user as the search results.

Threat Model: In our threat model, both data owners and data users are trusted. However, the cloud server is true, [1, 3, 4] but weird. Though data files are encrypted, the cloud server may try to get other sensitive information from user search requests while performing keyword-based search. So, the search should be carry out in a secure manner that allows data files to be securely retrieved while making interesting as little information as possible to the cloud. We consider the threat models as follow:

Know Ciphertext Model: The cloud server can only know the encrypted files, the secure index and the applied trapdoors. The cloud server can also identify and record the search results. The semantic meaning of this threat script is captured by the non-adaptive attack model [5].

Known Background Model: The cloud server knows excess background information in this model. The background refers to the information which can be gain from a comparable data-set. For example, the correlation relationship of two applied trapdoors. In this Model, the cloud server can use scale analysis to conclude the keyword specific information, which can be further combined with background information to recognize the keyword in a query at high possibility.

Design Goals

Support More Spelling Mistakes: Our multi-keyword fuzzy search scheme should hold up more spelling mistakes. For example, “applied Electronics” associated files should be found for a misspelled query “applid Electronics”, “appled Electronics”, “applide Electronics.

Privacy Guarantee: The cloud server should be precluded from obtaining additional information from the encrypted data files and the index.

No Predefined Dictionary: No predefined dictionary is a great offering of original scheme, so our scheme should not have predefined dictionary.

Support Updating: The same as original scheme, our scheme should carry dataset updating, such as file adding, file deleting and file modifying.

Ranked Results According to the Relevance Score: To make users more contented with search results, the return results should be ranked according to relevance score.

Efficiency and Accuracy: The effectiveness of our scheme should be same as the original scheme. And our scheme should be as precise as possible and keep high accuracy.

Preliminaries: Three important techniques are used in our design: Stemming algorithm, Bloom Filter, Locality-Sensitive Hashing (LSH).

Stemming Algorithm: A stemming algorithm [6] is a process of linguistic normalization, in which the alternative forms of a word are reduced to a common form. A "walker" for English, for example, should identify the string "cats" (and possibly "catlike", "catty" etc.) as based on the root "cat" and "walks", "walker", "walking", as based on "walk". A stemming algorithm minimize the words "talking", "talked" and "talker" to the root word, "talk". On the other hand, "rectify", "rectified", "rectifies", "rectifying" and "rectifis" diminish to the stem "rectifi" (illustrating the case where the stem is not itself a word or root). It is generally adopted in Information Retrieval systems to improve performance.

Bloom Filter: Bloom filter is a kind of data structure with very high efficiency. It makes use of the *m*-bit array to act as a pck and can find out whether an section belongs to the collection. It is initially set to 0 in all positions and for a given set $S = \{a1, a2, \bullet\bullet\bullet, an\}$, use l individualistic hash functions from $H = \{hi \mid hi\colon S \rightarrow m,\ 1 \leq i \leq l\}$ to insert an element a ◎ S into the Bloom filter by setting the positions to be 1.

Locality-Sensitive Hashing (LSH): LSH is an algorithm for resolving the approximate or exact Near Neighbor Search in high dimensional spaces. LSH hashes input items so that identical items are mapped to the same buckets with high probability.

Basic Idea of Our Scheme: We develop a novel method of keyword alteration based on the uni-gram. For incorrect spelling of one character, this method minimizes the Euclidean distance between the misspelled keyword and the exact keyword. Other than that, this method is also productive for other spelling mistakes. Additionally, we initiate the stemming algorithm to obtain the root of the word. Using this method, the keywords with the same root can also be enquired. We take the keyword weight into consideration in designing the ranked list. The files that are more apt to the keywords will have greater chances to become visible first on the list. We implement and assess our proposed approach using a real-world data set. The results reveal that our proposed scheme efficiently achieves high accuracy.

Multi-Keyword Fuzzy Search: One of the first works to label the problem of multi-keyword fuzzy search [4, 7] over encrypted data problem and did not need a predefined fuzzy set, mentioned to as MFSE. In MFSE, a keyword was first changed into a bi-gram set and the Euclidean distance was used to catch keywords likeness. The MFSE subsequently used LSH functions from the same hash family to produce the Bloom filter based index and query. Due to the nature of LSH, if the keyword was misspelled, it still could be hashed into the corresponding bits in the query vectors with high possible action. This method used the product which is inside of the index vector and the query vector as the relevance score between queries and documents. This scheme answered the problems of multi-keyword fuzzy search with high efficiency and accuracy. The most valuable result was that their scheme does not need the predefined fuzzy set.

Keyword Search Algorithm: In computer science, a search algorithm is an algorithm that recovers information stored within some data structure. These may contain linked lists, search trees, hash tables, arrays, etc., like storage methods. The matching search algorithm often depends on the data structure possibly searched. To query the data structure, searching also contain algorithms like SQL SELECT command. Search algorithms can be ordered by their mechanism of searching. Linear search algorithms check every record for the one related with a target key in a linear fashion. Binary, or half interval searches, often target the center of the search structure and divide the search space in half. Comparison search algorithms improve on linear searching by successively eliminating records based on comparisons of the keys until the target record is found and can be applied on data structures with a defined order. Digital search algorithms work based on the properties of digits in data structures that use numerical keys. At last, hashing directly maps keys to records build on a hash function. Searches outside of a linear search need that the data be sorted in some way.

Search functions are also assessed based on their complexity, or maximum theoretical run-time. Binary search functions, for example, have a maximum complexity of O(log(n)), or logarithmic time. This means that the maximum number of operations required to find the search target is a logarithmic function of the size of the search space.

Permutation Search: Permutation search is a useful tool that searches for terms that carry the set of keywords arranged in various sequences. This is a more limited and is particularly useful when you are trying to target a specific group of keywords. To use this feature, the keywords must be entered detached by **", "**. "n/a" in the search column focuses that the keyword has no searches registered in the Keyword Discovery database.

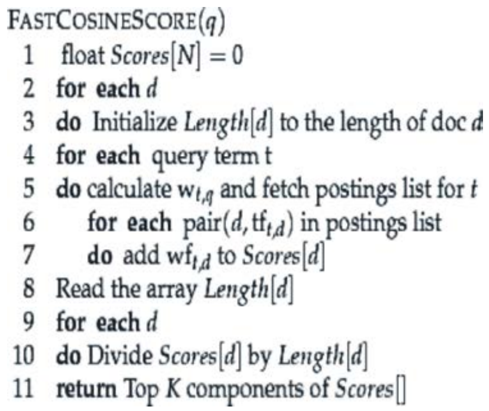
Scoring and Ranking: Some of the multi-keyword searchable symmetric encryption schemes bear only Boolean queries, i.e., a file either matches or does not match a query. Considering the large number of data users and documents in the cloud, it is essential to allow multi-keyword in the search query and get back documents in the order of their relevancy with the queried keywords.

Scoring is a normal way to weight the relevance. Based on the relevance score, files can then be ranked in either ascending or descending. Several models have been suggested to score and rank [8] files in information retrieval (IR) community.

Computing Vector Score: In a typical setting, we have a heap of documents each entitled by a vector, a free text query represented by a vector and a positive integer k. We seek the k documents of the group with the highest vector space scores on the given query. Typically, we search these k top documents in ordered by decreasing score. The array length contains the lengths (normalization factors) for each of the N- documents, w here as the array scores holds the scores for each of the documents.

When the scores are finally computed in step 9, all that remains in step 10 is to pick off the k-documents with the highest scores. The outermost loop beginning step 3 repeats the updating of scores, repeating over each query term-t in turn. In step 5 we calculate the weight in the query vector for term-t. Step 6-8update the score of each document by adding in the contribution from term-t. This sometimes known as term-at-a-time scoring or accumulation and the N elements of the array scores are therefore known as accumulators. For this purpose, it would appear necessary to store, with each postings entry, the weight wft, dof term-t in document-d.

A faster algorithm for vector space scores:



```
FASTCOSINESCORE($q$)
1  float $Scores[N] = 0$
2  for each $d$
3  do Initialize $Length[d]$ to the length of doc $d$
4  for each query term t
5  do calculate $w_{t,q}$ and fetch postings list for $t$
6     for each pair($d$, $tf_{t,d}$) in postings list
7     do add $wf_{t,d}$ to $Scores[d]$
8  Read the array $Length[d]$
9  for each $d$
10 do Divide $Scores[d]$ by $Length[d]$
11 return Top $K$ components of $Scores[]$
```

Efficient Scoring and Ranking: We begin by recapping the above algorithm. For a query such as q= jealous gossip, two observations are immediate:

- The unit vector has only two non-zero components.
- In the absence of any weighting for query terms, these non-zero components are equal - in this case, both equal 0.707.

For ranking the documents matching this query, we are focused in the relative (rather than absolute) scores of the documents in the collection. To this end, it suffices to calculate the cosine similarity from each document unit vector to (in which all non-zero components of the query vector are set to 1), rather than to the unit vector. For any two documents d1, d2. For any document d, the cosine resemblance is the weighted sum, over all terms in the query q, of the weights of those terms in d. This in turn can be evaluated by a postings intersection exactly as in the algorithm, with line 8 altered since we take wt,q to be 1 so that the multiply-add in that step becomes just an addition. We walk through the postings in the inverted index for the terms in q, assemble the total score for each document - very much as in processing a Boolean query, except we assign a positive score to each document that become visible in any of the postings being traversed. We continue an idf value for each dictionary term and a tf value for each postings entry. This scheme work out a score for every document in the postings of any of the query terms; the total number of such documents may be substantially smaller than N.

Given these scores, the final step before presenting outcomes to a user is to pick out the K-highest-scoring documents. While one could sort the complete set of scores, a better way is to use a heap to retrieve only the top K- documents in order. Where J-is the number of documents with non-zero cosine scores, establishing such a heap can be performed in 2J comparison steps, following which each of the K highest scoring documents can be ``read off'' the heap with log J comparison steps.

TRSE Design: Existing SSE schemes engage server-side ranking based on order preserving encryption to improve the efficiency of retrieval over encrypted cloud data. However, server-side ranking based on order-preserving encryption breaks the privacy of sensitive information, which is considered intransigence in the security-oriented third-party cloud computing scenario, i.e., security cannot be tradeoff for efficiency. To attain data privacy, ranking must be left to the user side. Traditional user-side schemes, however, load heavy computational haul and high communication overhead on the user side, due to the interaction between the server and the user including searchable index [1] return and ranking score computation. Thus, the user side ranking schemes are provoked by practical use.

A more server-siding scheme might be a better result to privacy issues. We suggest a new intractable encryption method, in which novel technologies in cryptography community and IR community are adapted, including homomorphic encryption and vector space model. In the proposed scheme, the data owner encrypts the searchable index [1] with homomorphic encryption. When the cloud server collects, query consist of multi-keyword, it computes the scores from the encrypted index stored on cloud and then deliver the encrypted scores of files to the data user. Then, the data user decrypts the scores and picks out the top-k highest-scoring files' identifiers to request to the cloud server. The retrieval takes a two-round transmission between the cloud server and the data user. We thus name the scheme as two round searchable encryption (TRSE) scheme, in which ranking is done at the user side while scoring computation is done at the server side.

Data Mining Algorithms: An algorithm in data mining (or machine learning) is a set of heuristics and calculations that generates a model from data. Algorithm first examines the data you provide, while creating a model, that looking for specific types of patterns or trends. The algorithm uses the results of this analysis over many iterations to seek the optimal parameters for creating the mining model. These parameters are then wooing across the entire data set to extract actionable patterns and detailed statistics.

The mining model that an algorithm generates from your data can take different forms, including:

- A set of clusters that say how the cases in a dataset are related.
- A decision tree that predicts an outcome and explains how different criteria affect that outcome.
- A mathematical model that predicts sales.
- A set of rules that describe how products are categorized together in a transaction and the probabilities that products are purchased together.

The algorithms provided in SQL Server Data Mining are the most popular, well-researched procedures of deriving patterns from data. To take one example, K-means clustering is one of the oldest clustering algorithms and is available mainly in many different tools and with many different executions and options. However, the implementation of K-means clustering used in SQL Server Data Mining was developed by Microsoft Research and then optimized for performance with Analysis Services. All the Microsoft data mining algorithms can be extensively practiced and are fully programmable, using the provided APIs. You can also automate the creation, training and retraining of models by using the data mining elements in Integration Services.

You can also use third-party algorithms that observe with the OLE DB for Data Mining specification, or generate custom algorithms that can be registered as services and then used within the SQL Server Data Mining framework.

C4.5 Algorithm: C4.5 is an algorithm used to produce a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification and for this reason, C4.5 is often mentioned to as a statistical classifier.

At each node of the tree, C4.5 selects the attribute of the data that most effectively splits its group of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain, difference in entropy. The attribute with the highest normalized information gain is considered to make the decision. The C4.5 algorithm then recurs on the smaller subsists.

This algorithm has a few base cases.

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

Security Intend Computational Encryption: To reduce the computational burden on user side, evaluating work should be at the server side, so we need an encryption scheme to guarantee the operability and security at the same time on server side. Homomorphic encryption [9] allows specific kinds of computations to be carried out on the corresponding cipher text. The outcome is the cipher text of the result of the same operations performed on the plaintext. That is, homomorphic encryption allows calculation of cipher text without knowing anything about the plaintext to get the accurate encrypted result. Although it has such a fine property, original fully homomorphic encryption scheme, which deals ideal lattices over a polynomial ring, is too complicated and inefficient for practical utilization. As an outcome of employing the vector space model to top-k retrieval, only addition and multiplication operations over integers are required to compute the relevance scores from the encrypted searchable index. Therefore, we can alleviate the original homomorphism in a full form to a simplified form that only supports integer operations, which permits more efficiency than the full form does.

Encoding/Decoding Algorithms: The Rijndael algorithm is a new generation symmetric block cipher that supports key sizes of 128, 192 and 256 bits, with data holded in 128-bit blocks - however, in excess of AES design criteria, the block sizes can mirror those of the keys. Rijndael uses a variable number of rounds, depending on key/block sizes, as follows:

9 rounds if the key/block size is 128 bits
11 rounds if the key/block size is 192 bits
13 rounds if the key/block size is 256 bits

Rijndael is a substitution linear transformation cipher, not needing a Feistel network. It employs triple discreet invertible uniform transformations (layers). Specifically, these are: Linear Mix Transform; Non-linear Transform and Key Addition Transform. Even before the first round, a simple key addition layer is performed, which adds to security. Thereafter, there are Nr-1 rounds and then the final round. The transformations form a State when begin but before completion of the entire process.

Related Work: The first practical searchable encryption scheme for searching on encrypted data and provided evidences of security for the resulting cryptosystems. A 2-layered encryption structure was used to find over the encrypted documents with a sequential scan [1]. A searchable symmetric encryption scheme to reach the keyword search. The main concept of their execution was building an upside-down index to store a list of mappings from keywords to the corresponding set of files that contain this keyword. However, it only supported single keyword search. A privacy-preserving multi-keyword ranked search scheme which used coordinate matching to perceive ranked search [2]. H. Li. [10] proposes a practical inverted index based public key searchable encryption scheme. Their scheme issue stronger security guarantee of the index and trapdoor privacy and is more systematic compared with the existing public-key searchable encryption schemes. [11] improved the scheme by reducing the index size. [7] Proposed a privacy-aware bed-tree model to help Fuzzy multi-keyword search, which designed the pre-defined phrases as the isolated keyword. Secure multi- keyword ranked search method based on vector space model (VSM). Their scheme used an MDB-tree as its index structure which improved the search complexity in a sense that the cloud server only needs to search the part of the tree [12]. Wang *et al.* [13] employed a score table to make the scheme support range query. Fu *et al.* [4] used the stemming algorithm to obtain the root of the query keyword so that the resulting method can find out more related files.

CONCLUSION

Here we enquire the problem of multi-keyword fuzzy ranked search over encrypted cloud data. We suggest a multi-keyword fuzzy ranked search scheme based on Wang *et al.*'s scheme. We develop a novel methodology of keyword transformation and introduce the stemming algorithm. With these two techniques, the proposed scheme can successfully handle more misspelling mistake. Moreover, our proposed scheme takes the keyword weight into deliberation during ranking. Like Wang *et al.*'s scheme, our proposed strategy doesn't need a predefined keyword set and hence enables efficient file update. We also give thorough security analysis and conduct experiments on real world data set, which predict the proposed strategy's possible of practical usage.

Future Works: When the user's query is a character string or sentence, we can extract the attributes of a sentence and then indicate the relationship between attributes and search though the attributes, it is called as semantic search. We failed to achieve the ideal state due to the keyword weight. We will develop a way to reflect the keyword weight and enable update. We will design a verifiable search scheme over encrypted cloud data. Nowadays, many works were motly concentrating on the cases of single data owner and hence not fruitful for multi-data owner. Note that multidate owner scheme has more practical significance.

REFERENCES

1. Goh, E.J., 2003. Secure indexes, in Proc. Cryptol. EPrint Arch. Oct. 2003, pp: 1-19.
2. Wang, C., N. Cao, M. Li, K. Ren and W. Lou, 2011. Privacy-preserving multi-keyword ranked search over encrypted cloud data, in Proc. IEEE INFOCOM, Apr. 2011, pp: 829-837.
3. Li, J., Q. Wang, C. Wang, N. Cao, K. Ren and W. Lou, 2010. Fuzzy keyword search over encrypted data in cloud computing, in Proc. IEEE INFOCOM, Mar. 2010, pp: 1-5.
4. Wang, J., X. Yu and M. Zhao, 2015. Privacy-preserving ranked multi-keyword fuzzy search on cloud encrypted data supporting range query, Arabian J. Sci. Eng., 40(8): 2375-2388.
5. Curtmola, R., J. Gray, S. Kamara and R. Ostrovsky, 2006. Searchable symmetric encryption: Improved definitions and efficient constructions, in Proc. CCS, 2006, pp: 79-88.
6. Porter, M.F., 1980. An algorithm for suffix stripping, Program, 14(3): 130-137.
7. Chuah, M. and W. Hu, 2011. Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data, in Proc. 31st Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW), Jun. 2011, pp: 273-281.
8. Xia, Z., X. Wang, X. Sun and Q. Wang, 2016. A secure and dynamic multikeyword ranked search scheme over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst., 27(2): 340-352, Feb. 2016, doi: 10.1109/TPDS.2015.2401003.
9. Boneh, D. and B. Waters, 2007. Conjunctive, subset and range quires on encrypted data, in Proc. 4th Theory Cryptogr. Conf., Amsterdam, Netherlands, Feb. 2007, pp: 535-554.
10. Li, H., Y. Yang, T.H. Luan, X. Liang, L. Zhou and X.S. Shen, 2016. Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data, IEEE Trans. Dependable Secure Comput., 13(3): 312-325, May/Jun. 2016.
11. Liu, C., L. Zhu, L. Li and Y. Tan, 2011. Fuzzy keyword search on encrypted cloud storage data with small index, in Proc. ICCCIS, Sep. 2011. pp: 269-273.
12. Sun, W., *et al.*, 2013. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, in Proc. 8th ASIACCS, 2013, pp: 71-82.
13. Fu, Z., J. Shu, X. Sun and D. Zhang, 2014. Semantic keyword search based on trie over encrypted cloud data, in Proc. 2nd Int. Workshop Security Cloud Comput., Kyoto, Japan, Jun. 2014, pp: 59-62.